April 1994

UILU-ENG-94-2218
DAC-48

Analog and Digital Circuits

Technology Decomposition for Low-power Synthesis

Rajendran Panda and Farid N. Najm

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) (DAC-48)	5. MONITORING ORGANIZATION REPORT NUMBER(S) MIP-9308426
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION National Science Foundation
6c. ADDRESS (City, State, and ZIP Code) 1308 W Main St Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 1800 G St NW Washington, DC 20550
8a. NAME OF FUNDING / SPONSORING ORGANIZATION National Science Foundation 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 1800 G St NW Washington, DC 20550	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Technology Decomposition for Low-Power Synthesis

12. PERSONAL AUTHOR(S)
Panda, Rajendran; Najm, Farid N.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	94 Apr 28	20

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	VLSI; synthesis; low-power; technology; decomposition; tree

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

(Attached)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete.*

Power minimization has recently become an important objective of automatic VLSI synthesis systems. Since the power in CMOS circuits is directly dependent on the extent of circuit switching activity, one important task of low-power synthesis is to minimize the total circuit activity. In this paper, we focus on the technology decomposition phase of the synthesis process, and present a novel low-power decomposition algorithm that makes use of circuit activity information. Our decomposition is based on selecting an ordering of the signals, depending on their activity and probability and the type of Boolean function decomposed. The algorithm is simple, fast (O(nlog n)), and yields very attractive power savings. Furthermore, it is applicable to both synchronous and asynchronous static circuits. The performance of the algorithm is evaluated through gate-level timing simulations of the decomposed circuits, under an assignable delay model. The results show power reductions of up to 56% for AND and OR trees compared to an arbitrary balanced tree decomposition.

Technology Decomposition for Low-Power Synthesis†

Rajendran Panda and Farid N. Najm

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

Power minimization has recently become an important objective of automatic VLSI synthesis systems. Since the power in CMOS circuits is directly dependent on the extent of circuit switching activity, one important task of low-power synthesis is to minimize the total circuit activity. In this paper, we focus on the technology decomposition phase of the synthesis process, and present a novel low-power decomposition algorithm that makes use of circuit activity information. Our decomposition is based on selecting an ordering of the signals, depending on their activity and probability and the type of Boolean function decomposed. The algorithm is simple, fast ($\mathcal{O}(n \log n)$), and yields very attractive power savings. Furthermore, it is applicable to both synchronous and asynchronous static circuits. The performance of the algorithm is evaluated through gate-level timing simulations of the decomposed circuits, under an assignable delay model. The results show power reductions of up to 56% for AND and OR trees compared to an arbitrary balanced tree decomposition.

---

† This work was supported in part by the National Science Foundation (NSF), under grant MIP-9308426.

Submitted to the IEEE International Conference on Computer-Aided Design, 1994.

1. Introduction

The dramatic decrease in feature size and the corresponding increase in the number of devices on a chip, combined with the growing demand for portable communication and computing systems, has made power consumption a major concern in VLSI design [8, 9]. Indeed, the Semiconductor Industry Association has identified low-power design techniques as a critical technological need [10]. Excessive power dissipation in integrated circuits not only discourages their use in a portable environment, but also causes overheating, which degrades performance and reduces chip lifetime. Thus there is a need for low-power design methodologies for VLSI circuits.

Recently, there has been increased interest in the *low-power synthesis* area [11–15], so that chips may be automatically designed to have low power dissipation. In addition to the traditional synthesis goals of high speed and small area, an additional objective of low power is now being considered.

In this paper, we focus on the *logic synthesis* [16] phase of the overall synthesis flow, and specifically on the *technology decomposition* step. In this step, logic gates from the minimized multi-level netlist are decomposed into smaller gates for which realizations are known to exist in the cell library. Technology decomposition and the companion step of technology mapping have also been the subject of recent research activity [1–3]. We will show that 50% reduction in power dissipation may be achieved by using a new decomposition algorithm that makes use of circuit switching activity information.

In the popular CMOS and BiCMOS technologies, the chip components (gates, cells) do not draw steady state power supply current, but rather draw current only when they make a logic transition. Thus the power-dissipation becomes highly dependent on the *switching activity* inside the circuit. Simply put, a more active circuit will consume more power. This complicates the low-power synthesis problem because the power becomes a moving target - it is input pattern dependent.

Input signals are generally unknown during the design phase because they depend on the system in which the chip will eventually be used. Furthermore, it is practically impossible to estimate the power by simulating the circuit for all possible inputs. We refer to this as a pattern-dependence problem and we get around it by using probabilities to describe the set of *all possible logic signals*, and then study the power resulting from the collective influence of all these signals. This formulation achieves a certain degree of *pattern-independence* that allows one to efficiently estimate and manipulate the power dissipation.

Specifically, we will use a measure of activity in digital circuits, called the *transition density* [5], that can be efficiently evaluated without requiring *exact* information about the primary input signals. The transition density at a node x in the circuit is the *average number*

of logic transitions per second at that node, denoted by $D(x)$. As part of this formulation, it is found [5] that the density $D(x)$ is directly related, through a simple linear expression, to the Boolean function at node x. Therefore, more judicious choices for internal functional blocks based on these relationships will, *by-design*, give less active and lower-power circuits. We will make use of these notions to develop a low-power technology decomposition algorithm.

The rest of this paper is organized as follows. Some background material is reviewed in section 2, and our new algorithm is presented in section 3. Experimental results are given in section 4, and section 5 contains a summary and conclusions.

2. Background

We focus on the popular design style of *synchronous sequential circuits.* In this circuit model, the design consists of latches or register banks driven by a common clock and combinational logic blocks that take inputs from some latches and feed their outputs to other latches. We also assume that the latches are edge-triggered, and that we have a CMOS or BiCMOS design style in which the circuit draws no steady state supply current. Instead, the circuit consumes power only when a gate or latch switches. The average power dissipation of the whole circuit can be broken down into the power consumed by the latches and that consumed by the combinational logic blocks.

Whenever the clock triggers the latches, some of them will transition and will draw power. Thus latch power is drawn in synchrony with the clock. The same is not true for gates inside the combinational logic. Even though the inputs to a combinational logic block are updated by the latches (in synchrony with the clock), the internal gates of the block may make several transitions before settling to their steady state values for that clock period.

These additional transitions have been called hazards or glitches. Although unplanned for by the designer, they are not necessarily design errors. Only in the context of low-power design do they become a nuisance, because of the additional power that they dissipate. It has been observed [13] that this additional power dissipation is typically 20% of the total power, but can easily be 70% of the total power in some cases such as combinational adders. We have observed that in one case of a multiplier circuit, some nodes make as many as 20 transitions before settling down to a final state. This component of the power dissipation is computationally expensive to estimate, because it depends on the timing relationships between signals inside the circuit. Consequently, most previous work in this area has ignored this issue. We will refer to this elusive component of power as the *toggle power.*

As an instance of the technology decomposition problem, consider that the minimized multi-level circuit resulting from logic synthesis contains an 8-input AND gate. If the gate library contains only 2-input gates, then this gate must be decomposed into seven 2-input

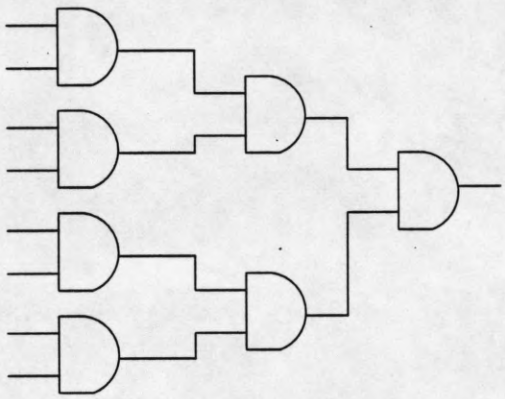


Figure 1. A balanced tree decomposition of an 8-input AND.

gates that, for instance, might be arranged as shown in Fig. 1.

Thus the result of technology decomposition is a tree of identical gates. In the case of Fig. 1, the tree is *balanced*, which is usually desirable for minimizing the delay. As we will see later on, this turns out to be undesirable for power minimization, and a skewed or unbalanced tree is preferred.

In accordance with the above discussion, when the original gate is at the primary inputs, the tree inputs switch in synchrony and carry at most a single transition per clock cycle. In this case, we will say that the gate/tree is operating in synchronous mode. Otherwise, the tree inputs do not switch in synchrony and they may carry multiple transitions per cycle, in which case the gate/tree is said to be operating asynchronously. Thus there are two challenging issues in technology decomposition for low power: handling the toggle power, and allowing both synchronous and asynchronous operation.

In order to allow some degree of pattern-independence, the signals in the tree are usually described with probabilities. The following definitions are commonly used:

Definition 1. (signal probability): *The signal probability at a node is defined as the average fraction of clock cycles in which the steady state value of the node is a logic high.*

Definition 2. (transition probability): *The transition probability at a node is defined as the average fraction of clock cycles in which the steady state value of the node is different from its initial value.*

It is important to note that both these probability measures are unaffected by the circuit internal delays. Indeed, they remain the same even if a zero-delay timing model is used. When this is done, however, the toggle power is automatically excluded from the analysis and synchronous operation is automatically assumed for all gates. This is a serious shortcoming of some previously proposed technology decomposition and mapping techniques, as we will point out below.

If a zero delay model is used and the transition probabilities (denoted by $P_t(x)$) computed, then the power can be computed as:

$$P_{av} = \frac{1}{2T_c} V_{dd}^2 \sum_{i=1}^{n} C_i P_t(x_i) \tag{1}$$

where T_c is the clock period, C_i is the total capacitance at node x_i, and n is the total number of nodes in the circuit. Since this assumes at most a single transition per clock cycle, then this is actually a *lower bound* on the true average power.

Another aspect of this problem that usually requires approximations is the signal independence issue. In practice, the signals at the tree inputs may be correlated so that, for instance, two of them may never be simultaneously high. It is computationally too expensive to compute these correlations, so that the inputs to the tree are usually assumed to be *independent.* We refer to this as a *spatial independence* assumption. Another independence issue is whether the values of the same signal in two consecutive clock cycles are independent or not. If assumed independent, then the transition probability can be easily obtained from the signal probability according to $P_t(x) = 2P_s(x)P_s(\overline{x})$, where $P_s(x)$ denotes signal probability. We refer to this as a *temporal independence* assumption.

The technology decomposition technique that we will propose makes only a *spatial independence* assumption. Otherwise, it accounts for toggle power and allows for synchronous or asynchronous operation. We do not make a temporal independence assumption.

2.1 Previous work

This being a very recent problem, there is only one reference [3] that discusses low-power technology decomposition *per se.* Otherwise, references [1] and [2] discuss the related problem of low-power technology mapping and assume that the decomposition step has already been performed. In [1], a *zero-delay assumption* is used to simplify power estimation during the technology mapping algorithm; thus the toggle power is excluded. In [2], transition probabilities, coupled with a *zero-delay model*, are used to estimate power, based on (1), during technology mapping; thus the toggle power is also excluded. Furthermore, both *temporal and spatial independence* are assumed.

The results in [3] include a low-power technology decomposition algorithm. The authors assume *zero-delay* and *temporal and spatial independence*, and consider both dynamic (e.g., DOMINO CMOS) and static circuits. Dynamic circuits are simpler to deal with because all nodes are initially precharged, and the transition probability becomes identical to the signal probability. Thus they observe that, based on (1), the power in the tree is proportional to the sum of the signal probabilities at all the tree nodes. Minimizing this sum leads to a Huffman's algorithm formulation [6] to generate a minimum power tree.

For static circuits, which is what we are considering in this paper, the situation is more complicated. In [3], a *zero-delay assumption* that allows the power to be expressed, as in (1), in terms of the transition probabilities, which are computed assuming *temporal and spatial independence.* Since the propagation of transition probabilities in the tree does not constitute a quasi-linear function, Huffman's algorithm does not apply in this case. Nevertheless, [3] uses Huffman's algorithm anyway as a heuristic, using the transition probabilities as weights, to try and minimize (1).

The procedure that we will propose in the section 3 makes *only* a spatial independence assumption. We *do not* make the zero-delay assumption and *do not* assume temporal independence. In order to maintain computational efficiency, however, the procedure is heuristic and does not guarantee an absolutely minimum power decomposition. To assess the accuracy, we measured the power with a deterministic timing-simulation based method [7]. Our results show that the technique works very well in all cases; we always obtained a lower power tree, compared to a conventional balanced tree decomposition.

2.2 A review of the transition density formulation

Our technology decomposition algorithm is based on the *transition density* formulation [5]. The transition density at node x is the average number of transitions per second at node x, denoted $D(x)$. Formally:

Definition 3. (transition density) *If a logic signal $x(t)$ makes $n_x(T)$ transitions in a time interval of length T, then the transition density of $x(t)$ is defined as:*

$$D(x) \triangleq \lim_{T \to \infty} \frac{n_x(T)}{T} \tag{2}$$

The density provides an effective measure of switching activity in logic circuits. If the density at every circuit node is made available, the overall average power dissipation in the circuit can be easily computed as:

$$P_{av} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{n} C_i D(x_i) \tag{3}$$

In a synchronous circuit, with a clock period T_c, the relationship between transition density and transition probability is $D(x) \geq P_t(x)/T_c$, where equality occurs in the zero-delay case.

For completeness, we briefly review the density propagation procedure. Let $P(x)$ denote the *equilibrium probability* [5] of a logic signal $x(t)$, defined as the *average fraction of time that the signal is high.* Formally:

Definition 4. (equilibrium probability) *If $x(t)$ is a logic signal, then its equilibrium probability is defined as:*

$$P(x) \triangleq \lim_{T \to \infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} x(t) dt \tag{4}$$

In contrast to the signal probability, the equilibrium probability depends on the circuit internal delays since it describes the signal behavior over time and not only its steady state behavior per clock cycle. In the zero-delay case, the equilibrium probability reduces to the signal probability.

We also recall the concept of *Boolean difference*: if y is a Boolean function that depends on x, then the Boolean difference of y with respect to x is defined as:

$$\frac{\partial y}{\partial x} \triangleq y|_{x=1} \oplus y|_{x=0} \tag{5}$$

where $\oplus$ denotes the exclusive-or operation. It was shown in [5] that, if the inputs x_i to a Boolean module are *spatially independent*, then the density of its output y is given by:

$$D(y) = \sum_{i=1}^{n} P\left(\frac{\partial y}{\partial x_i}\right) D(x_i) \tag{6}$$

The *simplicity* of this expression makes for very efficient CAD implementations. Given the probability and density values at the primary inputs of a logic circuit, a single pass over the circuit, using (6), gives the density at every node. In order to compute the Boolean difference probabilities, we also propagate the equilibrium probabilities $P(x_i)$ from the primary inputs throughout the circuit, as described in [5].

As an example, consider the simple case of a 2-input logic AND gate: $y = x_1 x_2$. In this case, $\partial y/\partial x_1 = x_2$ and $\partial y/\partial x_2 = x_1$, so that:

$$D(y) = P(x_2)D(x_1) + P(x_1)D(x_2) \tag{7}$$

In more complex cases, where f is a general Boolean function, Binary Decision Diagrams can be used [5] to carry out the computation (6).

In order to use this approach, the user needs to specify the density and probability at the circuit primary inputs. To some extent, this makes the approach not completely pattern-independent. However, this information is usually much more readily available to designers than specific input patterns are. For instance, it is relatively easy for a designer to estimate average input frequencies, say by looking at test vector sets, or simply by assuming some nominal average frequency based on the clock frequency. In any case, since average power dissipation *is* pattern-dependent, and since it is directly related to the average frequency, it would not make sense to expect to estimate power without some information about the average frequency at the circuit inputs. The transition density formulation is simply an effective way of using this information to find the activity at every circuit node.

The generality of the definition (2) and the simplicity of the result (6) make the transition density an excellent candidate for solving the low-power synthesis problem. It's application to the technology decomposition problem is presented in the next section.

3. Decomposition Algorithm

Depending on the tree type, different decomposition algorithms will be applied. We will study the three cases of an AND tree, an OR tree and an XOR or XNOR tree. Implementations in terms of other gate types can be usually deduced from these. In each case, we will reach a formulation of the problem that can be solved by the application of Huffman's algorithm [6], using an appropriate choice of node weights. We assume that the inputs to the tree are *spatially independent* and that the transition density and equilibrium probability are provided for each tree input. These values may be obtained by using the density propagation procedure [5] (reviewed in section 2.2), or by some other means, such as [7].

3.1 AND tree decomposition

In this case, we use a heuristic procedure that works by using Huffman's algorithm to reduce the "overall values" of equilibrium probability in the tree. This is in contrast to [3] where the overall *transition* probabilities were minimized in the case of static circuits.

We start with some general observations about the AND function. If $y = x_1 x_2$ then we know from (6):

$$D(y) = P(x_2)D(x_1) + P(x_1)D(x_2) \tag{8}$$

This can be written:

$$\frac{D(y)}{P(x_1)P(x_2)} = \frac{D(x_1)}{P(x_1)} + \frac{D(x_2)}{P(x_2)} \tag{9}$$

From which it follows that:

$$\frac{D(y)}{P(y)} = \frac{D(x_1)}{P(x_1)} + \frac{D(x_2)}{P(x_2)} \tag{10}$$

In general, if we let $R(x) = D(x)/P(x)$, then the above result can be generalized so that if $y = x_1 x_2 \cdots x_n$, we have:

$$R(y) = R(x_1) + \cdots + R(x_n) \tag{11}$$

To motivate the decomposition procedure, suppose an n-input AND gate is to be decomposed into a tree built of 2-input ANDs, called a binary tree. The input signals are leaf nodes of the tree, and each 2-input AND becomes a tree internal node. A binary tree with n leaves always has $(n-1)$ internal nodes. Let $x_i, i = 1, \ldots, n$ denote the leaves of the tree and $a_j, j = 1, \ldots, (n-1)$ denote the internal nodes. The average power dissipated in the tree is given by:

$$P_{av} = \frac{1}{2} V_{dd}^2 \sum_{j=1}^{(n-1)} C_j D(a_j) \tag{12}$$

where $D(a_j)$ is the transition density at the output of the gate corresponding to node a_j and C_j is its output capacitance. The gate output capacitances are not known exactly at this

point. They can only be determined exactly during the technology mapping step. However, with an appropriately chosen C_{max}, one can write:

$$P_{av} \leq \frac{1}{2} V_{dd}^2 C_{max} \sum_{j=1}^{(n-1)} D(a_j) = \frac{1}{2} V_{dd}^2 C_{max} \sum_{j=1}^{(n-1)} R(a_j) P(a_j) \tag{13}$$

If we denote the set of leaves of the sub-tree rooted at a_j by $\{x_i, i \in \mathcal{I}_j\}$, then:

$$R(a_j) = \sum_{i \in \mathcal{I}_j} R(x_i) \qquad \text{and} \qquad P(a_j) = \prod_{i \in \mathcal{I}_j} P(x_i) \tag{14}$$

so that:

$$P_{av} \leq \frac{1}{2} V_{dd}^2 C_{max} \sum_{j=1}^{(n-1)} \left(\sum_{i \in \mathcal{I}_j} R(x_i) \right) \left(\prod_{i \in \mathcal{I}_j} P(x_i) \right) \tag{15}$$

In order to minimize the average power, we would like to choose a tree decomposition that reduces the value of the upper bound in (15). Since $P(x_i) \leq 1$ then the last term in (15) (the product of $P(x_i)$ terms) will generally be a small number. Indeed, if we choose a tree with a lot of depth in which the leaves with smallest probabilities are placed far from the root (so that they appear in as many product terms as possible), then we should be able to reduce the upper bound appreciably due to the very small product terms generated. This suggests that the probabilities $P(a_j) = \prod_{i \in \mathcal{I}_j} P(x_i)$ are the key quantities that one should manipulate. In contrast, the $R(x_i)$ terms are part of a sum, rather than a product, so that they may not be as critical in reducing the upper bound.

It might seem strange that the power should be more sensitive to the leaf probabilities, rather than their densities. This, however, should not be surprising because low probabilities can block incoming transitions. This insight into the problem is possible due to the density propagation theorem [5] given by (6) which, when applied to the tree root y, gives:

$$D(y) = P(y) \times R(y) = \prod_{i=1}^{n} P(x_i) \times \sum_{i=1}^{n} \frac{D(x_i)}{P(x_i)} \tag{16}$$

It is clear from this that increasing the leaf probabilities by a factor $k > 1$ results in a factor of $k^{(n-1)}$ increase in the density at y, while a similar increase of leaf transition densities results only in k times the original density.

Coming back to the minimization strategy suggested above, we can make it somewhat more formal, at the risk of working with a looser bound, as follows:

$$\text{If } R_{max} \triangleq \max_{i=1,\ldots,n} (R(x_i)) \qquad \text{then} \qquad P_{av} \leq \frac{1}{2} V_{dd}^2 C_{max} R_{max} \sum_{j=1}^{(n-1)} |\mathcal{I}_j| P(a_j) \tag{17}$$

This upper bound still points to the same general minimization strategy: reduce the overall values of the $P(a_j)$ terms, especially those closer to the root (since $|\mathcal{I}_j|$ is large for these nodes). This again suggests that low probability leaves be kept far from the root, so that the internal tree probabilities are reduced as much as possible. A systematic way of doing this is to use Huffman's algorithm [6] which can be seen to be relevant to this problem if we make one final simplification of the upper bound, based on $|\mathcal{I}_j| \leq n$:

$$P_{av} \leq \frac{1}{2} V_{dd}^2 C_{max} R_{max} n \sum_{j=1}^{(n-1)} P(a_j) \tag{18}$$

The last term in this upper bound (the summation) can be minimized by a Huffman's algorithm that minimizes the sum of all tree node probabilities (including the leaves) using the node equilibrium probabilities as *weights*. It is easy to verify that the *weight function* $P(xy) = P(x)P(y)$ is quasi-linear [17] so that Huffman's algorithm can indeed be used and is optimal:

Start with a list of the leaf nodes, sorted by their probabilities. Combine the lowest two nodes on the list as the two inputs of an AND gate. Remove both nodes from the list and insert the output of this AND gate in the list, maintaining a sorted list. Repeat this until the list is exhausted.

Since $P(xy) < \min(P(x), P(y))$ then it's easy to see that this basic algorithm will result in a simple *chain*, an extreme case of an unbalanced tree, in which the leaf probabilities *increase monotonically* as one moves closer to the root, as shown inf Fig. 2.

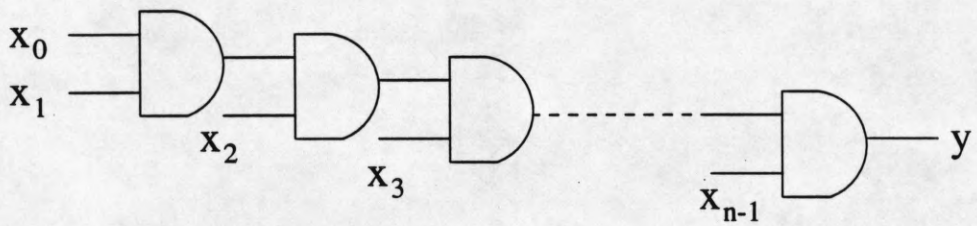
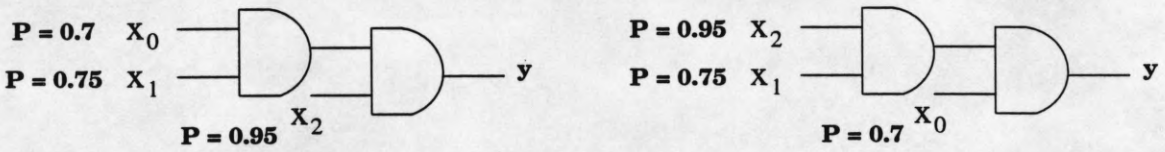


Figure 2. A chain-structured tree.

It is interesting to note that [3] arrived at the same Huffman's algorithm solution as above in the case of *dynamic* circuits, but not for static circuits. For static circuits, [3] uses a different Huffman's algorithm method in which the *transition probabilities* are used as weights. This is a consequence of their use of the temporal independence and zero-delay assumptions. These assumptions lead to a situation where the density is completely determined by the (signal) probability. In practice, however, the signal probabilities only give a *lower bound* $2P_s(x)P_s(\overline{x})/T_c$ on the density, as pointed out in section 2. Additional transitions (toggles) are often generated in practice, due for example to unequal delay paths.

The simple counter example shown in Fig. 3 illustrates a case where the algorithm of [3] leads to a chain that is different from ours and which consumes more power once the density is increased above its theoretical lower bound. The justification for increasing the density is to allow for temporal dependence and the possibility of toggles due to a non-zero delay model. The two chains are the result of decomposition of $y = x_0x_1x_2$ where the probabilities of the inputs are as shown in the figure. To measure the power in the two chains, we performed long timing simulation runs [7] using an accurate delay model that includes fanout capacitance and inertial delay, as described in more detail in section 4.



Decomposition using our algorithm Decomposition based on [3]

Figure 3. A counter example.

We then assumed this circuit to be part of a larger design with a clock frequency of 80 MHz and measured the power for various densities $D(x_0)$, $D(x_1)$, and $D(x_2)$ ranging from the theoretical lower bound $2P_s(x)P_s(\overline{x})/T_c$ at each node to 10 times the clock frequency. Our decomposition performed better, consuming 11% less power on average. In another experiment, we held $D(x_0)$ and $D(x_2)$ at their theoretical lower bounds of $2P_s(x)P_s(\overline{x})/T_c$, and changed $D(x_1)$ from 1 to 20 times its theoretical lower bound. Again our results were better, consuming 10%–20% less power (average of 15%).

To summarize, as we will show by experimental results in the next section, our proposed Huffman's algorithm based on the equilibrium probabilities works very well for static circuits, without making a zero-delay assumption or a temporal independence assumption. It is also easy to show that our algorithm applies equally well to dynamic circuits, confirming the results of [3] for those circuits.

Finally, we should add that the above results can be easily generalized and the same algorithm can handle trees with multi-input (greater than 2) gates as well.

3.2 OR tree decomposition

In this case, it can be shown that the results are almost identical to the AND tree case. The only difference is that $P(x_i)$ and $P(a_j)$ are replaced by $P(\overline{x}_i)$ and $P(\overline{a}_j)$, respectively. Thus the minimized OR tree is a chain in which the leaf probabilities *decrease monotonically* as one moves closer to the root.

3.3 Handling delay constraints

The above AND and OR decompositions result in a maximum number of levels in the tree,

which may be unacceptable, for instance, when the tree is on the critical path of the circuit. A good way to solve this problem in general is to modify the simple algorithm to take a specified *delay constraint* into account. Therefore, we propose a modified decomposition algorithm that tries to unbalance the tree as much as possible so long as it meets the delay constraint. At every level of the tree, the algorithm decides how many of the available fan-ins should be assigned to internal nodes, based on the number of available levels, unassigned leaves, and fan-ins in the present level.

Given the number of allowed levels in the tree, as well as the number of inputs, input probabilities and the fan-in limit of gates to be used, the algorithm proceeds as follows:

(1.) Initialize leaf nodes and store them in a INPUT-LIST, sorted by probability.

(2.) Determine the number of internal nodes (or gates) needed in the decomposed circuit: if there are n_i items in INPUT-LIST, and the fan-in limit is m, then the number of internal nodes is $p = \lceil (n_i - 1)/(m-1) \rceil$. Create and store the p internal nodes in a NODE-LIST.

(3.) Remove a node from NODE-LIST and designate it as the root. Decrement the number of available levels by 1.

(4.) Determine the number of available fan-ins: (available fan-ins) = (no. of internal nodes in the previous level) $\times$ (fan-in limit). Determine the minimum number of the above fan-ins that need to be connected to the internal nodes of the next level: if l is the no. of available levels, m is the fan-in limit, f is the total fan-in available in the current level, and n_i is the no. of items in INPUT-LIST, then, the minimum no. of internal nodes at the next level, x, should be chosen as the least value satisfying $x^l + (f - x) \geq n_i$, i.e., $x = \lceil (n_i - f)/(l-1) \rceil$.

(5.) Connect x input nodes from NODE-LIST to x fan-ins and connect $(f - x)$ leaves taken from INPUT-LIST to the remaining fan-ins. Decrement the no. of available levels by 1.

(6.) Repeat steps 4 and 5 until INPUT-LIST is empty.

It's clear that the algorithm is very fast, with a time complexity of $\mathcal{O}(n \log n)$, where n is the number of inputs. Furthermore, if the list of inputs is already sorted to begin with, then it is only linear time complexity, $\mathcal{O}(n)$.

3.4 XOR and XNOR tree decomposition

If $z = x \oplus y$ (XOR) or $z = x \overline{\oplus} y$ (XNOR), then it's clear from (6) that in either case $D(z) = D(x) + D(y)$. This is because an input to an XOR or XNOR gate has no controlling state that can disable the gate and block incoming transitions. This considerably simplifies the problem and leads to a formulation that can, again, be solved by Huffman's algorithm.

The average tree power is:

$$P_{av} = \frac{1}{2} V_{dd}^2 \sum_{j=1}^{(n-1)} C_j D(a_j) \tag{19}$$

Again, using some appropriate capacitance C_{max}, we can write:

$$P_{av} \leq \frac{1}{2} V_{dd}^2 C_{max} \sum_{j=1}^{(n-1)} D(a_j) \tag{20}$$

Since, in this case, the density of a tree node is simply the sum of the densities of its children, then the density can itself be used as a *weight function* in using Huffman's algorithm to minimize the sum in (20):

Start with a list of the leaf nodes, sorted by their densities. Combine the lowest two nodes on the list as the two inputs of an XOR or XNOR gate. Remove both nodes from the list and insert the output of this XOR or XNOR gate in the list, maintaining a sorted list. Repeat this until the list is exhausted.

In this case, we have found that the decomposition algorithm leads to a tree that is close to balanced.

4. Experimental Results

We evaluated the performance of our algorithm by estimating the power consumed in a large number of AND, OR and XOR nodes of different sizes, that were decomposed into low-power trees by our algorithm. The decomposition was carried out for several sets of input probability and density values, that were generated randomly. In each case, we also constructed the conventional balanced tree decomposition, and estimated the power consumption of the balanced trees. The performance of the algorithm was evaluated by comparing the power consumption of the low-power tree with the power consumption of the balanced tree.

4.1 Power Estimation

The power estimation for the decomposed circuits was done using a statistical estimator, called MED. MED [7] (acronym for Mean Estimator of Density) uses a statistical estimation technique to compute the individual node transition densities and the total power consumption. The circuit is described in a gate-level netlist format, along with the *equilibrium probability* and *transition density* values of its inputs. Using these values and a random generator, MED generates logic input waveforms, with which it performs a timing simulation of the circuit. The transition density at every node and the total power dissipation of the circuit are determined from the data taken from several samples. The simulation is event-driven, based on an assignable delay timing model, which is scaled by the fanout capacitance. The timing model also handles inertial delay, so that pulses that are too short compared to the gate inertial delay are recognized as glitches and no corresponding output event is generated. The simulation can be done either in the synchronous (all tree input events are in synchrony)

or the asynchronous mode. The desired accuracy of the results and a confidence factor can be specified to the simulator by the user. The results presented below were obtained at 5% error and at 95% confidence level. The decomposed circuits were evaluated in both, synchronous and asynchronous, modes to verify that the algorithm works well in both cases.

4.2 Results

The graph in Fig. 4(a) compares the power consumption of different decompositions of a 20-input AND node into 2-input nodes. Likewise, Fig. 4(b) is for a 20-input OR node. The balanced trees for these nodes have 5 levels and the chain-like low power trees have 19 levels. The points in the graph between these two extremes correspond to the trees that were constructed by the algorithm taking the delay constraint into consideration. The chain-like trees result in as much as 50% reduction in power, considering the balanced tree power as the reference. The graph shows that the algorithm is *well behaved*, meaning that in most regions the slope of the above curve is negative. There are very few points, like when increasing the number of levels from 6 to 7 (in the above graphs), where the power increases very little, nevertheless remaining much below the balanced tree power level. It is worth pointing out that even without introducing any additional delay, the algorithm results in a 20% reduction in power, compared to the a balanced tree based on an arbitrary input ordering. This is due to the better ordering of the leaves in the balanced tree which is not full.

Figs. 5 and 6 present results of decomposition of smaller size nodes, 10-input and 5-input respectively. The power-delay behavior in these cases remains similar to that of the 20-input node. The 10-input OR case shows an attractive power reduction of 56%.

Figs. 7 and 8 show that the algorithm performs uniformly well even with a coarser pattern, which is 3-input gates in these cases. Indeed, the power savings are substantial when the number of levels in the tree are only one or two more than the minimum.

A very important feature of our algorithm, that can be seen in all the above plots, is that significant reduction in power happens early-on, long before the tree has been deformed into a chain. Indeed, it is seen that small departures from a balanced tree structure produce the largest reductions in power. Another important point to note is that, the algorithm works well in both synchronous and asynchronous modes, so that whether the arrival times at the tree inputs are equal or not, the algorithm still performs well.

Finally, table I below summarizes the results of decomposing XOR nodes based on the input densities. Since our decomposition (which is exact in the XOR/XNOR case) leads to trees that are close to balanced, the reduction in power is small compared to an arbitrary balanced tree. It is important to note, however, that even when our decomposition does not increase the number of levels in the tree, it still manages to reduce the power somewhat due to the better ordering of the inputs that it generates.

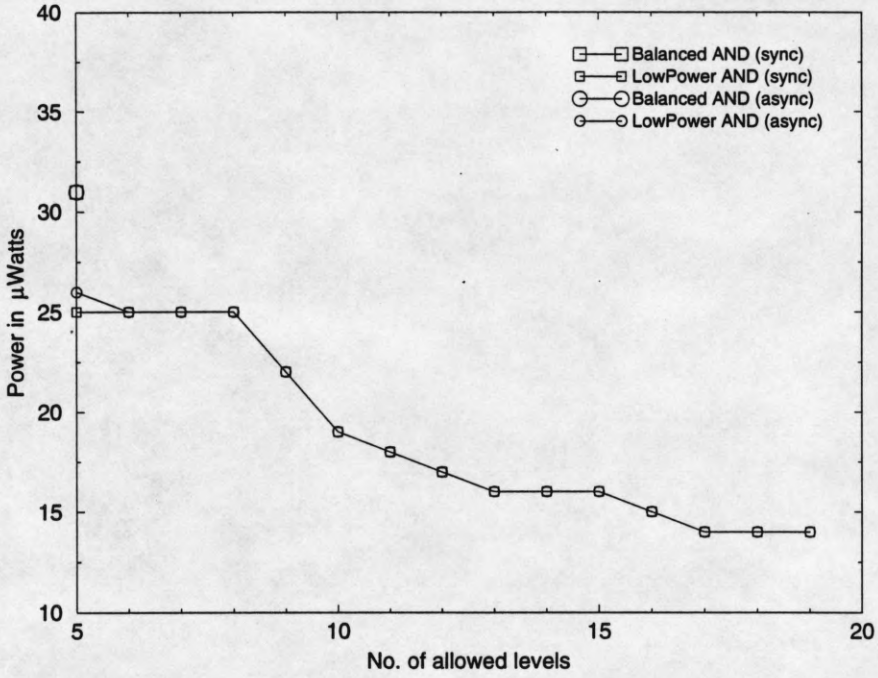


Figure 4(a). Power performance of low-power trees implementing a 20-input AND function, using 2-input gates.

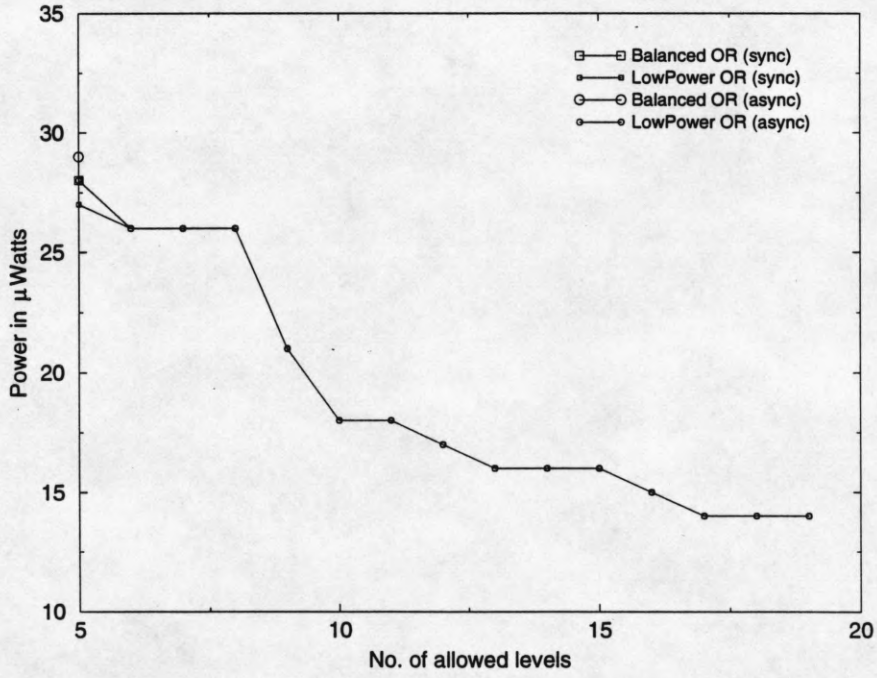


Figure 4(b). Power performance of low-power trees implementing a 20-input OR function, using 2-input gates.

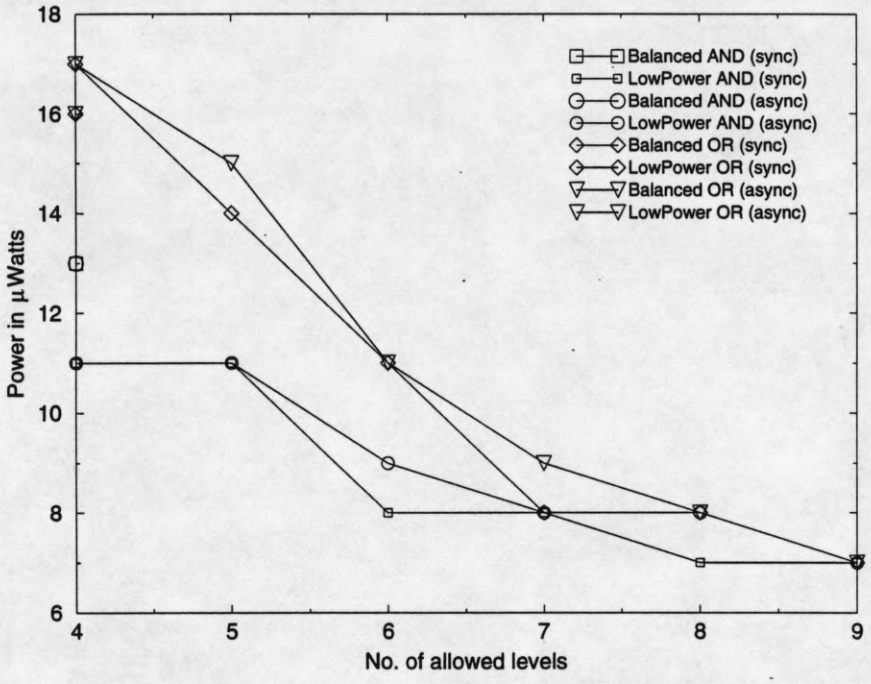


Figure 5. Power performance of low-power trees implementing 10-input AND and OR functions, using 2-input gates.

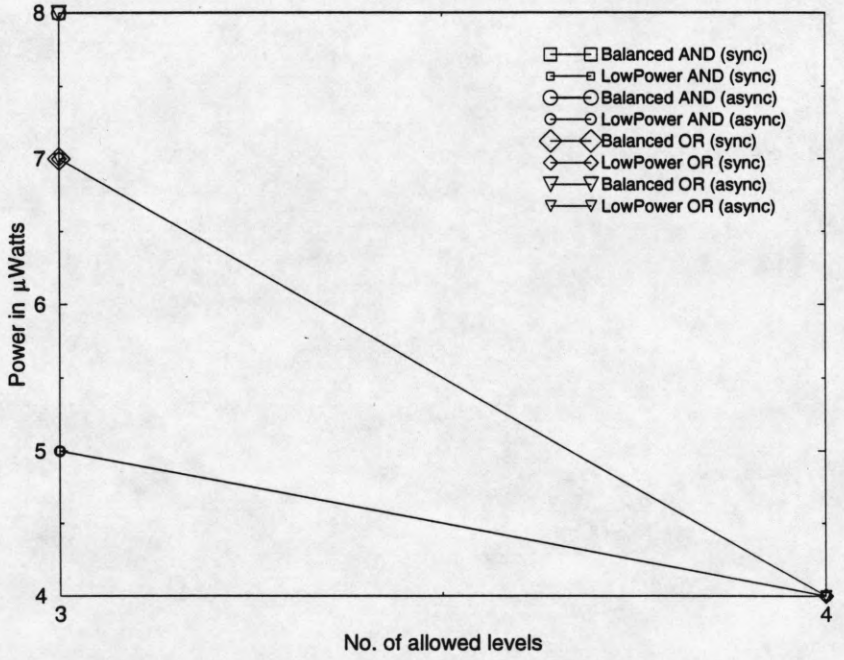


Figure 6. Power performance of low-power trees implementing 5-input AND and OR functions, using 2-input gates.

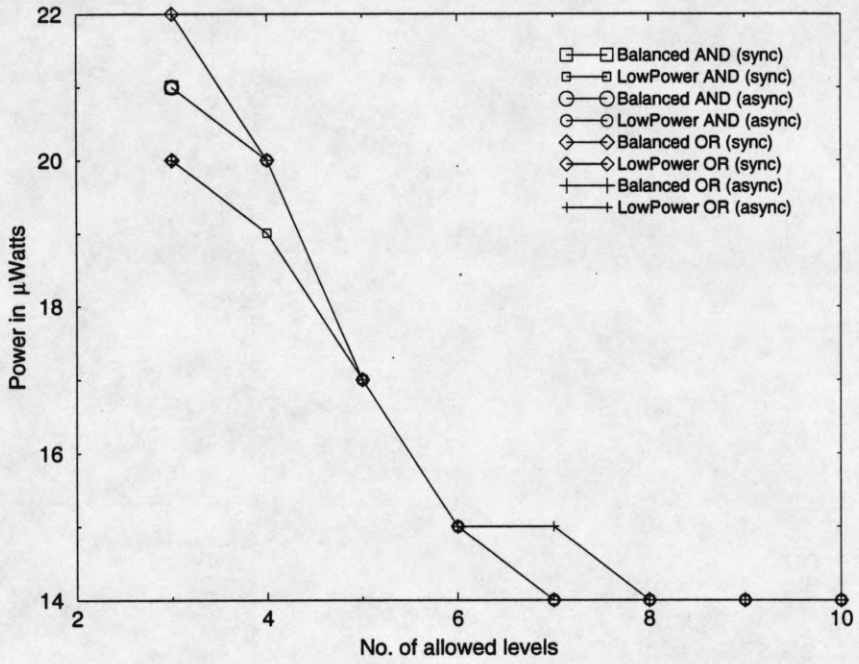


Figure 7. Power performance of low-power trees implementing 20-input AND and OR functions, using 3-input gates.

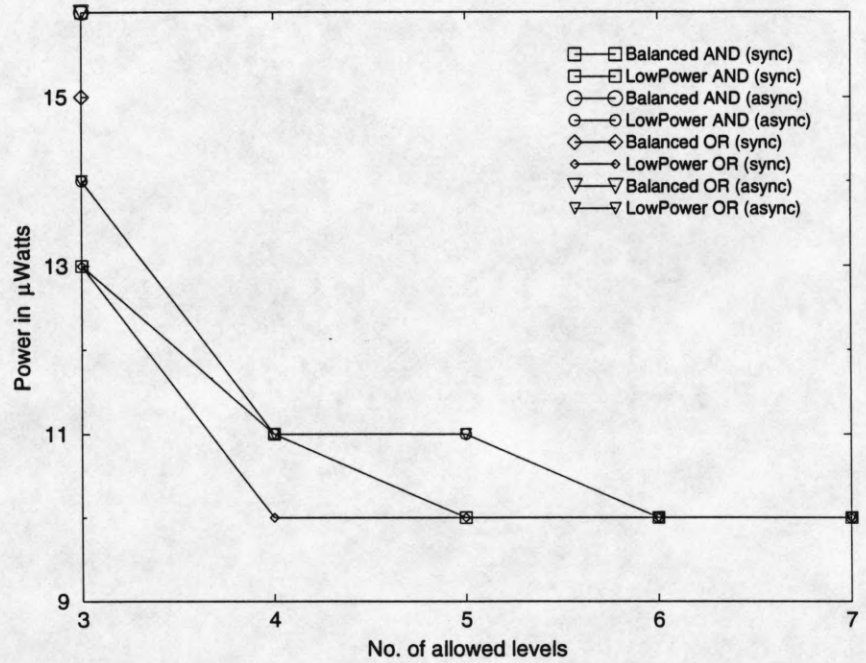


Figure 8. Power performance of low-power trees implementing 15-input AND and OR functions, using 3-input gates.

TABLE I

RESULTS OF DECOMPOSITION OF XOR NODES

#inputs	#levels		power in μW for sync. mode			power in μW for async. mode		
	bal	low-power	bal	low-power	% reduction	bal	low-power	%reduction
5	3	3	16.89	16.35	3.2	17.96	16.70	7.0
10	4	4	42.06	39.59	5.9	46.36	44.44	4.1
15	4	5	63.80	62.47	2.1	74.86	71.43	4.6
20	5	6	95.51	93.42	2.2	116.09	111.62	3.9

5. Summary and Conclusions

We have presented an algorithm for low-power technology decomposition, required to perform automatic synthesis of VLSI circuits. The novel feature of our approach is that we make use of the circuit activity information, based on the transition density formulation. Signal probability and density are used to select a proper ordering of the input signals of the decomposed tree, and the proper tree decomposition that provides low-power. Compared to the arbitrary balanced tree case, 9–56% power reduction was observed for AND and OR trees, and 1–7% reduction for XOR and XNOR trees.

References

[1] V. Tiwari, P. Ashar, and S. Malik, "Technology Mapping for Low Power," *Proc. 30th ACM/IEEE Design Automation Conference*, pp. 74–79, Dallas, TX, June 14–18, 1993.

[2] B. Lin and H. de Man, "Low-Power Driven Technology mapping under Timing Constraints," *International Workshop on Logic Synthesis '93 Workshop Notes*, pp. 9a-1–9a-16, 1993.

[3] C-Y Tsui, M. Pedram, and A. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," *30th ACM/IEEE Design Automation Conference*, pp. 68–73, Dallas, TX, June 14–18, 1993.

[4] MIS Source Code.

[5] F. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 310–323, Feb. 1993.

[6] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098-1101, September 1952.

[7] M. Xakellis and F. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," *31st ACM/IEEE Design Automation Conference*, 1994.

[8] R. W. Brodersen, A. Chandrakasan, S. Sheng, "Technologies for personal communications," *1991 Symp. on VLSI circuits*, Tokyo, Japan, pp. 5–9, 1991.

[9] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, April 1992.

[10] Workshop Working Group Reports, Semiconductor Industry Association, pp. 22–23, Nov. 17–19, 1992, Irving, Texas.

[11] K. Roy and S. Prasad, "SYCLOP: Synthesis of CMOS logic for low power applications," *IEEE International Conference on Computer Design*, pp. 464–467, 1992.

[12] A. P. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Brodersen, "HYPER-LP: A system for power minimization using architectural transformations," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 300–303, Santa Clara, CA, November 8–12, 1992.

[13] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," IEEE/ACM International Conference on Computer-Aided Design, pp. 402–407, Santa Clara, CA, November 8–12, 1992.

[14] F. Dresig, P. Lanches, O. Rettig, and U. G. Baitinger, "Simulation and reduction of CMOS power dissipation at logic level," *European Design Automation Conference*, pp. 341–346, 1993.

[15] K. S. Lowe and P. G. Gulak, "Gate sizing and buffer insertion for optimizing performance in power constrained BiCMOS circuits," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 216–219, Santa Clara, CA, November 7–11, 1993.

[16] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis," *Proceedings of the IEEE*, vol. 78, no. 2, pp. 264–300, February 1990.

[17] D. S. Parker, Jr., "Conditions for the optimality of the Huffman algorithm," *SIAM Journal of Computing*, vol. 9, no. 3, pp. 470–489, August 1980.